In the Claims

1. (Previously presented) A method in a computer system of executing a first sequence of modules in a first task, said first sequence of modules linked to one another and having at least one sequence of execution, comprising:

a. storing in each of said first sequence of modules a skip value indicating a next module in said first sequence of modules to execute;

b. executing a first module of said first sequence of said modules; and

c. executing said next module of said first sequence of modules indicated by the skip value stored in a currently executed module, skipping any module between the currently executed module and said next module, wherein each module of said first sequence of modules comprises at least one digital signal processing data structure.

2. (Previously presented) The method of claim 1 wherein the skip value comprises the integer N which indicates that execution should skip to the N+1th module following execution of the currently executed module in the first sequence of modules.

3. (Previously presented) The method of claim 2 wherein a skip value of N less than zero stored in the currently executed module indicates that execution of the first sequence of modules should terminate after completion of execution of the currently executed module.

4. (Original) The method of claim 1 further comprising executing elements a-c on a second sequence of modules, said second sequence of modules being a part of a second task.

5. (Cancelled)

6. (Original) The method of claim 1 further comprising performing a skip action created on the previous iteration of the first sequence of modules.

7. (Currently amended) The method of claim 1 wherein the skip value stored in each module in said first sequence of modules is modifiable by ~~a~~ the respective module ~~indicated by the skip value~~.

8. (Previously presented) The method of claim 1 wherein the skip value stored in each module in said first sequence of modules is modifiable by a host associated with the first task.

9. (Previously presented) A method of controlling execution flow of a first task comprising a first sequence of executable modules in a processing system by storing in each of said executable modules a skip count indicating a next module in said first sequence of modules to execute, said skip count comprising an integer N which indicates that execution should skip to the N+1th module following execution of a currently executed module in the first sequence of executable modules, skipping any module between the currently executed module and the N+1th module, a skip count of N less than zero stored in the currently executed module indicating that execution of the first sequence of modules should terminate after completion of execution of the currently executed module, wherein each module comprises at least one digital signal processing data structure.

10. (Currently amended) A method performed by a processor of controlling the flow of execution of a first set of executable modules sequentially associated with one another comprising:

a. executing a first module in said first sequence of modules;

b. determining a skip value ~~associated with~~stored in said first module, said skip value indicating a module to execute subsequent to execution of said first module; and

c. proceeding to execute a subsequent module in said first set of executable modules indicated by said skip value ~~associated with~~stored in a currently executed module, skipping any modules between the currently executed

module and the subsequent module, wherein each module comprises at least one digital signal processing data structure.

11. (Original) The method of claim 10 wherein the skip value comprises the integer N, and the subsequent module is the N+1th module following the first module in said first sequence of modules.

12. (Currently amended) The method of claim 11 wherein a skip value of N less than zero ~~associated with~~stored in said first module indicates that execution of said first sequence of modules should terminate after completion of execution of said first module.

13. (Currently amended) The method of claim 10 further comprising:

~~wherein~~ storing the skip value of each module of said first sequence of modules ~~is stored~~ in ~~a datum associated with~~ said each module.

14. (Previously presented) The method of claim 10 wherein the said skip value in each of said modules is modifiable by a host.

15. (Currently amended) The method of claim 10 wherein the said skip value in each of said modules is modifiable by ~~each of~~ the respective modules ~~in said first sequence of modules~~.

16. (Previously presented) An apparatus for executing a first sequence of modules in a first task, said first sequence of modules linked to one another and having at least one sequence of execution, comprising:

a. means for storing in each of said first sequence of modules a skip value indicating a next module in said first sequence of modules to execute;

b. means for executing a first module of said first sequence of said modules; and

c. means for executing said next module of said first sequence of modules indicated by the skip value stored in a currently executed module, skipping

any modules between the currently executed module and the next module, wherein each module comprises at least one digital signal processing data structure.

17. (Currently amended) An apparatus for controlling the flow of execution of a first set of executable modules sequentially associated with one another comprising:

a. means for executing a first module in said first sequence of modules;

b. means for determining a skip value ~~associated with~~stored in said first module, said skip value indicating a module to execute subsequent to execution of said first module; and

c. means for proceeding to execute a subsequent module in said first set of executable modules indicated by said skip value ~~associated with~~stored in a currently executed module, skipping any modules between the currently executed module and the subsequent module, wherein each module comprises at least one digital signal processing data structure.

18. (Previously presented) A method of controlling the execution sequence of a series of modules by a processor, each of said modules associated with one another, comprising:

a. executing the first in said series of modules;

b. determining a skip value N stored in said first in said series of said modules;

c. if the skip value N stored in said first module is less than zero, then terminating the execution of said series of modules;

d. else if the skip value N stored in said first module is greater than or equal to zero then proceeding to a N+1th module in said series of said modules, skipping any module between the first module and the N+1th module, wherein each of said modules comprises at least one digital signal processing data structure.

19. (Previously presented) A method in a computer system of performing a first sequence of modules in a first task, said first sequence of modules linked to one another and having at least one sequence of execution, comprising:

a. storing in a first module of said first sequence of modules a skip value N representing a subsequent module in said first sequence of modules to execute, said skip value N comprising either:

i. an integer less than zero indicating that said first module is a last executable module to be executed in said first sequence of modules;

ii. an integer greater than or equal to zero indicating that said process should proceed to said N+1th module subsequent to said first module in said first sequence of said modules, skipping any module between the first module and the N+1th module;

b. executing the first of said first sequence of said modules; and

c. executing the subsequent module in said first sequence of said modules indicated by said skip value stored in a currently executed module, wherein each module of said first sequence of modules comprises at least one digital signal processing data structure.

56. (Currently amended) The method of claim 18, further comprising:

~~wherein~~ storing the skip value ~~is stored~~ in said first in said series of said modules.

57. (Currently amended) A machine-readable medium having executable instructions to cause a machine to control a flow of execution of a first set of executable modules sequentially associated with one another comprising:

executing a first module in said first set of executable modules, wherein each module comprises at least one digital signal processing data structure; and

executing a subsequent module in said first set of executable modules indicated by a skip value ~~associated with~~stored in a currently executed module, skipping any modules between the currently executed module and the subsequent module.

58. (Currently amended) The machine-readable medium of claim 57 further comprising:

storing the skip value of each module of said first set of executable modules in ~~a datum associated with~~ each module of said first set of modules.

59. (Previously presented) The machine-readable medium of claim 57 further comprising:

modifying the skip value in each of said modules by a host.

60. (Currently amended) The machine-readable medium of claim 57 further comprising:

modifying the skip value in each of said modules by ~~a~~ the respective module ~~indicated by the skip value~~.

61. (Previously presented) The machine-readable medium of claim 57, wherein the first set of modules forms a first task and further comprising:

performing the instructions against a second sequence of modules, said second sequence of modules being part of a second task.

62. (Previously presented) The machine-readable medium of claim 57 further comprising:

performing a skip action created on a previous iteration of the first set of executable modules.

63. (Previously presented) The machine-readable medium of claim 57, wherein the skip value comprises an integer N and the subsequent module is the N+1th module following the first module in said first sequence of executable modules, and wherein a value of N less than zero associated with the currently executed module indicates that execution of said first set of executable modules should terminate after completion of execution of the currently executed module.